Fast and reliable object pose estimation from line correspondences

Stéphane Christy, Radu Horaud

Fast and Reliable Object Pose Estimation from Line Correspondences *

Stéphane Christy and Radu Horaud

GRAVIR-IMAG & INRIA Rhône-Alpes
655 avenue de l'Europe
38330 Montbonnot Saint-Martin FRANCE

Abstract. In this paper, we describe a fast object pose estimation method from 3-D to 2-D line correspondences using a perspective camera model. The principle consists in iteratively improving the pose computed with an affine camera model (either weak perspective or paraperspective) to converge, at the limit, to a pose estimation computed with a perspective camera model. Thus, the advantage of the method is to reduce the problem to solving a linear system at each iteration step. The iterative algorithms that we describe in detail in this paper can deal with non coplanar or coplanar object models and have interesting properties both in terms of speed and rate of convergence.

1 Introduction and background

The problem of object pose from 2-D to 3-D correspondences has received a lot of attention for the last years. The perspective camera model has associated with it, in general, non linear object pose computation techniques. This naturally leads to non linear minimization methods which require some form of initialization [3,7,9] . If the initial "guess" is too faraway from the true solution then the minimization process is either very slow or it converges to a wrong solution. Moreover, such resolution techniques does not take into account the simple link that exists between the perspective model and its linear approximations.

Recently, Dementhon and Davis [2] proposed a method for determining the pose of a 3-D object with respect to a camera from 3-D to 2-D point correspondences. The method consists of iteratively improving the pose computed with a weak perspective camera model to converge, at the limit, to a pose estimation computed with a perspective camera model. Although the perspective camera model involves, in general, non linear techniques, the advantage of the method is to reduce the problem to solving, at each iteration step, a linear system – due to the weak perspective model.

In [5], Horaud and al. have extended the method of Dementhon and Davis to paraperspective by establishing the link between paraperspective and perspective models for object pose computation from point correspondences.

* This work has been supported by "Société Aérospatiale" and by DGA/DRET.

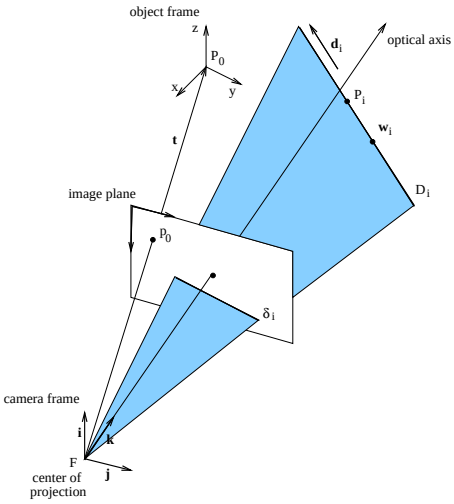


Fig. 1. This figure shows the general setup. The point $\mathbf{P}_0$ is the reference point of the object frame and its projection is $\mathbf{p}_0$. A 3-D line is represented by a reference point $\mathbf{w}_i$ and a direction vector $\mathbf{d}_i$ expressed in the object frame ($\mathbf{w}_i \perp \mathbf{d}_i$). The projection of the 3-D line D_i is δ_i.

However, it may be useful in some context to be able to deal with other primitives than points – lines for example. The advantage of lines is that they can be more accurately located than points and the robustness of their matching with respect to occlusions. For real time applications, it's generally easier to consider lines, particularly to do the tracking along a sequence of images, and the matching with a 3-D model. In some cases where edges are difficult to extract, the points may be erroneous due to missing segments and thus bring wrong matches which could be solved if we consider the lines described by each edge, without the end points.

In this article, we describe two linear methods to compute object pose from 3-D to 2-D line correspondences. We establish the link between affine camera models (weak perspective and paraperspective), and the perspective model in the case of lines. The proposed algorithms are iterative and suppose, at each iteration step, an affine camera model in order to have linear equations, but converge to the perspective camera model.

2 Camera model

We denote by $\mathbf{P}_i$ a 3-D point belonging to a 3-D line i represented by a reference point $\mathbf{w}_i$ and a direction $\mathbf{d}_i$. We have: $\mathbf{P}_i = \mathbf{w}_i + \lambda_i \mathbf{d}_i$. The origin of the 3-D frame is $\mathbf{P}_0$. A point $\mathbf{P}_i$ projects onto the image in $\mathbf{p}_i$ with camera coordinates x_i and y_i and we have:

$$x_i = \frac{\mathbf{i} \cdot \mathbf{P}_i + t_x}{\mathbf{k} \cdot \mathbf{P}_i + t_z} \qquad y_i = \frac{\mathbf{j} \cdot \mathbf{P}_i + t_y}{\mathbf{k} \cdot \mathbf{P}_i + t_z} \tag{1}$$

We divide both the numerator and the denominator of eqs. (1) by t_z. We introduce the following notations:

$$\mathbf{I} = \frac{\mathbf{i}}{t_z} \quad \mathbf{J} = \frac{\mathbf{j}}{t_z} \quad x_0 = \frac{t_x}{t_z} \quad y_0 = \frac{t_y}{t_z} \quad \text{and} \quad \varepsilon_i = \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z}$$

$\mathbf{I}$ and $\mathbf{J}$ represent the first two rows of the rotation matrix scaled by the z-component of the translation vector, and x_0 and y_0 are the camera coordinates of $\mathbf{p}_0$ which is the projection of $\mathbf{P}_0$.

We may now rewrite the perspective equations as:

$$x_i = \frac{\mathbf{I} \cdot \mathbf{P}_i + x_0}{1 + \varepsilon_i} \qquad y_i = \frac{\mathbf{J} \cdot \mathbf{P}_i + y_0}{1 + \varepsilon_i} \tag{2}$$

with

$$\mathbf{P}_i = \mathbf{w}_i + \lambda \mathbf{d}_i \quad \text{and} \quad \varepsilon_i = \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z} = \frac{\mathbf{k} \cdot \mathbf{w}_i}{t_z} + \lambda \frac{\mathbf{k} \cdot \mathbf{d}_i}{t_z}$$

By introducing

$$\eta_i = \frac{\mathbf{k} \cdot \mathbf{w}_i}{t_z} \quad \text{and} \quad \mu_i = \frac{\mathbf{k} \cdot \mathbf{d}_i}{t_z}$$

we finally have:

$$\varepsilon_i = \eta_i + \lambda \mu_i$$

3 From weak perspective to perspective

In this section, we derive the pose equations in the case of line correspondences and we show how to find the perspective solution from incremental weak perspective approximations.

Weak perspective assumes that the object points or lines lie in a plane parallel to the image plane passing through the origin of the object frame, i.e., $\mathbf{P}_0$.

The equation of the line i in an image may be written as:

$$a_i x + b_i y + c_i = 0 \tag{3}$$

a_i, b_i and c_i are the parameters of the image line. If $\mathbf{p}_i$ belongs to this line, we obtain, by substituting x and y (eqs. (2)) in equation (3):

$$a_i \frac{\mathbf{I} \cdot (\mathbf{w}_i + \lambda \mathbf{d}_i) + x_0}{1 + \eta_i + \lambda \mu_i} + b_i \frac{\mathbf{J} \cdot (\mathbf{w}_i + \lambda \mathbf{d}_i) + y_0}{1 + \eta_i + \lambda \mu_i} + c_i = 0$$

or by eliminating the denominator and grouping terms:

$$a_i \mathbf{I} \cdot \mathbf{w}_i + b_i \mathbf{J} \cdot \mathbf{w}_i + a_i x_0 + b_i y_0 + c_i (1 + \eta_i) + \lambda (a_i \mathbf{I} \cdot \mathbf{d}_i + b_i \mathbf{J} \cdot \mathbf{d}_i + c_i \mu_i) = 0$$

This equation is verified for all points $\mathbf{P}_i$ of the 3-D line, i.e., for all values of λ. So we obtain two equations:

$$a_i \mathbf{I} \cdot \mathbf{w}_i + b_i \mathbf{J} \cdot \mathbf{w}_i + a_i x_0 + b_i y_0 + c_i (1 + \eta_i) = 0 \tag{4}$$

$$a_i \mathbf{I} \cdot \mathbf{d}_i + b_i \mathbf{J} \cdot \mathbf{d}_i + c_i \mu_i = 0 \tag{5}$$

In these equations, $\mathbf{w}_i$ and $\mathbf{d}_i$ (which represent the 3-D line) are known, and a_i, b_i, c_i are also known (they may be computed from the image). The unknowns are the 3-D vectors $\mathbf{I}$ and $\mathbf{J}$ (which represent the orientation of the camera), the projection of the reference point (x_0, y_0), η_i and μ_i (the perspective effect). However, we don't need any point correspondences, but only line correspondences.

In order to solve the pose problem, we notice that if we take $\eta_i = \mu_i = 0$, eqs. (4) and (5) are similar to the equations obtained for a weak perspective camera model [1,2,5]. We therefore conclude that:

– Whenever the η_i and μ_i are fixed (not necessarily null) the pose equations (4) and (5) become linear in $\mathbf{I}$, $\mathbf{J}$, x_0 and y_0.
– It is possible to solve eqs. (4) and (5) *iteratively* by successive linear approximations.

In this case the pose algorithm starts with a weak perspective camera model and computes an approximated pose. This approximated pose is improved iteratively as follows:

1. For all i, $i \in \{1...n\}$, $\eta_i = \mu_i = 0$;
2. Solve the overconstrained linear system of equations (4) and (5) which provides an estimation of vectors $\mathbf{I}$ and $\mathbf{J}$, and the projection of the origin of the frame (x_0, y_0).
3. Compute the position and orientation of the object frame with respect to the camera frame:

$$t_z = \frac{1}{2}\left(\frac{1}{\|\mathbf{I}\|} + \frac{1}{\|\mathbf{J}\|}\right) \quad t_x = x_0\, t_z \quad t_y = y_0\, t_z \quad \mathbf{i} = \frac{\mathbf{I}}{\|\mathbf{I}\|} \quad \mathbf{j} = \frac{\mathbf{J}}{\|\mathbf{J}\|} \quad \mathbf{k} = \mathbf{i}\times\mathbf{j}$$

4. Enforce the orthogonality constraint, i.e., compute a *true* rotation matrix $\mathbf{R}_\perp$ which verifies: $\mathbf{R}_\perp \begin{pmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{pmatrix}^T = \mathbf{I}_{3\times 3}$
5. For all i, compute: $\eta_i = \mathbf{k}\cdot\mathbf{w}_i/t_z$ and $\mu_i = \mathbf{k}\cdot\mathbf{d}_i/t_z$. If the η_i and μ_i computed at this iteration are equal to the η_i and μ_i computed at the previous iteration then stop the procedure, otherwise go to step 2.

4 From paraperspective to perspective

In this section, we provide a generalization of the above algorithm to deal with the paraperspective case.

We consider again the perspective equations (2) and we make explicit the paraperspective approximation of the camera model. We obtain [1,5,8]:

$$x_i = \frac{\mathbf{I}_p \cdot \mathbf{P}_i}{1+\varepsilon_i} + x_0 \quad \text{and} \quad y_i = \frac{\mathbf{J}_p \cdot \mathbf{P}_i}{1+\varepsilon_i} + y_0 \tag{6}$$

with:

$$\mathbf{I}_p = \frac{\mathbf{i} - x_0\,\mathbf{k}}{t_z} \quad \text{and} \quad \mathbf{J}_p = \frac{\mathbf{j} - y_0\,\mathbf{k}}{t_z} \tag{7}$$

By substituting x and y (eqs. (6)) in equation (3) and by grouping terms:

$$\begin{aligned} &a_i\mathbf{I}_p\cdot\mathbf{w}_i + b_i\mathbf{J}_p\cdot\mathbf{w}_i + (a_ix_0 + b_iy_0 + c_i)(1+\eta_i) \\ &+ \lambda[a_i\mathbf{I}_p\cdot\mathbf{d}_i + b_i\mathbf{J}_p\cdot\mathbf{d}_i + (a_ix_0 + b_iy_0 + c_i)\mu_i] = 0 \end{aligned}$$

which yields, as above:

$$a_i\mathbf{I}_p\cdot\mathbf{w}_i + b_i\mathbf{J}_p\cdot\mathbf{w}_i + (a_ix_0 + b_iy_0 + c_i)(1+\eta_i) = 0 \tag{8}$$

$$a_i\mathbf{I}_p\cdot\mathbf{d}_i + b_i\mathbf{J}_p\cdot\mathbf{d}_i + (a_ix_0 + b_iy_0 + c_i)\mu_i = 0 \tag{9}$$

It is worthwhile to notice that when all the η_i and μ_i are null, the perspective equations above become identical to the paraperspective equations obtained with a paraperspective camera model.

The iterative algorithm is similar to the one described in the previous section. The only difference is step 2: we now compute $\mathbf{I}_p$ and $\mathbf{J}_p$ (instead of $\mathbf{I}$ and $\mathbf{J}$) by solving an overconstrained linear system of equations (8) and (9). Then (step 3), $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are deduced from $\mathbf{I}_p$ and $\mathbf{J}_p$ [1,5,8].

5 Solving the linear equations

Both the weak perspective and paraperspective iterative algorithms need to solve an overconstrained linear system of equations, namely eqs. (4), (5) (weak perspective) and eqs. (8), (9) (paraperspective). In matrix form these equations can be written as:

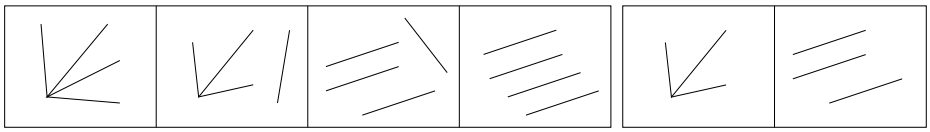
$$\underbrace{\mathbf{A}}_{2n\times 8}\ \underbrace{\mathbf{x}}_{8\times 1} = \underbrace{\mathbf{b}}_{2n\times 1} \tag{10}$$

– More explicitly, we have in weak perspective case:

$$\begin{pmatrix} \dots & \dots & \dots & \dots \\ a_i\mathbf{w}_i^T & b_i\mathbf{w}_i^T & a_i & b_i \\ a_i\mathbf{d}_i^T & b_i\mathbf{d}_i^T & 0 & 0 \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \mathbf{I} \\ \mathbf{J} \\ x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} \dots \\ -c_i(1+\eta_i) \\ -c_i\mu_i \\ \dots \end{pmatrix}$$

– In paraperspective case,

$$\begin{pmatrix} \dots & \dots & \dots & \dots \\ a_i\mathbf{w}_i^T & b_i\mathbf{w}_i^T & a_i(1+\eta_i) & b_i(1+\eta_i) \\ a_i\mathbf{d}_i^T & b_i\mathbf{d}_i^T & a_i\mu_i & b_i\mu_i \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \mathbf{I}_p \\ \mathbf{J}_p \\ x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} \dots \\ -c_i(1+\eta_i) \\ -c_i\mu_i \\ \dots \end{pmatrix}$$



(a) (b)

Fig. 2. Configurations of image lines which defeat the algorithm for non coplanar object (a) and coplanar object (b).

5.1 Non coplanar object lines

The linear system has 8 unknowns and thus must have at least 8 independent equations. Each line gives us 2 equations, one depending of the translation $\mathbf{w}_i$ and one depending of the direction $\mathbf{d}_i$. We must have at least 4 independent interpretation planes, i.e. we must have a subset of 4 lines in the image with no more than 2 lines intersect in the same point (see figure 2a). Then, the solution for $\mathbf{x}$ is simply given by: $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$.

5.2 Coplanar object lines

If the object lines are coplanar then the rank of $\mathbf{A}$ is only 6 and the above solution cannot be envisaged anymore. Let's consider the plane formed in this case by the object lines and let $\mathbf{u}$ be the unit vector orthogonal to this plane. Vectors $\mathbf{I}$ and $\mathbf{J}$ (and equivalently $\mathbf{I}_p$ and $\mathbf{J}_p$) can be written as a sum of a vector belonging to this plane and a vector perpendicular to this plane, namely: $\mathbf{I} = \mathbf{I}_0 + \alpha\,\mathbf{u}$ and $\mathbf{J} = \mathbf{J}_0 + \beta\,\mathbf{u}$.

By substituting these expressions for $\mathbf{I}$ and $\mathbf{J}$ into eqs. (4) and (5) (respectively eqs. (8) and (9) for paraperspective), and by noticing that $\mathbf{w}_i \cdot \mathbf{u} = 0$ and $\mathbf{d}_i \cdot \mathbf{u} = 0$, the system may be rewritten as: $\mathbf{A}'\mathbf{x}' = \mathbf{b}'$.

The rank of $\mathbf{A}$ is 6 if we have at least 3 independent interpretation planes, i.e. we must have a subset of 3 lines in the image with no more than 2 lines intersect in the same point (see figure 2b). The vector $\mathbf{u}$ is orthogonal to $\mathbf{I}_0$ and $\mathbf{J}_0$, thus $\mathbf{u} \cdot \mathbf{I}_0 = 0$ and $\mathbf{u} \cdot \mathbf{J}_0 = 0$ are two more independent equations and the matrix $\mathbf{A}'$ is of rank 8. Thus we obtain solutions for $\mathbf{I}_0$ and $\mathbf{J}_0$: $\mathbf{x}' = (\mathbf{A}'^T\mathbf{A}')^{-1}\mathbf{A}'^T\mathbf{b}'$.

In order to estimate $\mathbf{I}$ and $\mathbf{J}$ (and equivalently $\mathbf{I}_p$ and $\mathbf{J}_p$) one is left with the estimation of two scalars, α and β.

- In weak perspective case, Oberkampf, Dementhon, and Davis [6] provided a solution using the constraints $\|\mathbf{I}\| = \|\mathbf{J}\|$ and $\mathbf{I} \cdot \mathbf{J} = 0$.
- In paraperspective case, Horaud and al. [5] provided a solution using the constraints onto $\mathbf{I}_p$ and $\mathbf{J}_p$.

5.3 Enhancements

Solving the linear system: One may notice that in the case of the weak perspective iterative algorithm, the matrix $\mathbf{A}$ (or $\mathbf{A}'$ for coplanar model) doesn't depend of η_i and μ_i. Thus the pseudo inverse of this matrix remains the same at each iteration step, and therefore needs to be computed only once. It's not the case for paraperspective.

Normalization: The two iterative algorithms solve a linear system at each iteration step. For numerical reasons, we have to normalize each line of the matrix system because the norm of the vectors $\mathbf{d}_i$ and $\mathbf{w}_i$ may be very different between each line. Moreover, for a coplanar model, the vector $\mathbf{u}$ must also be normalized.

6 Experiments

In the first class of experiments, we study the performance of the iterative weak and paraperspective algorithms on simulated images. Two types of performances are studied:

- the precision of pose as a function of position and orientation of the object with respect to the camera in the presence of image (pixel) noise, and
- the convergence of the iterative pose algorithms.

The intrinsic camera parameters are $u_c = v_c = 256$, $\alpha_u = \alpha_v = 1000$. Gaussian noise with standard deviation $\sigma = 1$ pixel has been added to the image measurements and there are 500 trials for each experiments (the object is rotated at 500 random orientations). Finally, the 3-D model represents a simulated house and is made of 18 lines.

Convergence: In all these cases, both algorithms have converged in 100% of the configurations, between 3 and 5 iterations.

Execution time: With no optimization at all, execution time is 0.04 second per iteration on a UltraSparc 1/170 (for 18 lines).

Comparison: According to figure 3, the two iterative algorithms have the same accuracy. The only difference between weak perspective and paraperspective is the number of iterations which is generally one or two less with the paraperspective algorithm (see figure 4).

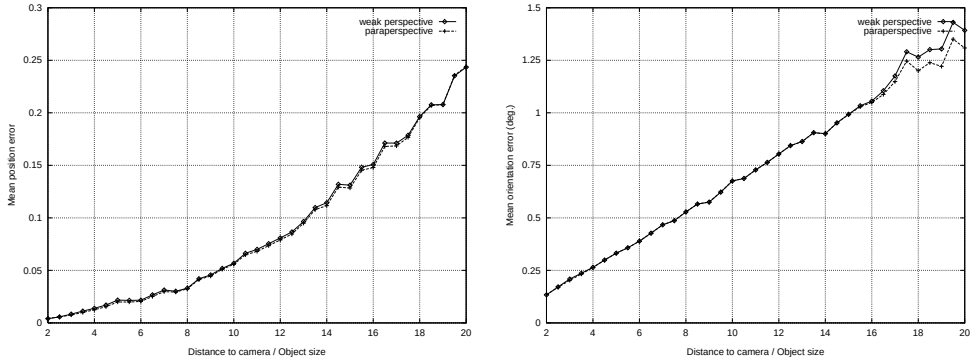


Fig. 3. On the left, error in position as a function of depth in presence of image Gaussian noise – On the right, error in orientation as a function of depth in presence of image Gaussian noise.

An example of application of object pose is the grasping of a polyhedral object by a parallel-jaw gripper (figure 5a). Both the image and the model are described by a network of straight lines and junctions which are matched using a method described in [4]. The following figures represent the quality of object pose computation obtained with two different algorithms. The 3-D object has been reprojected into the image to see the error of the pose.

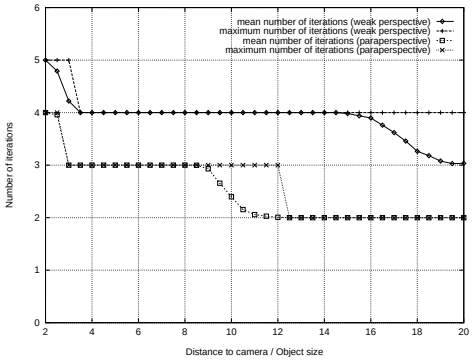


Fig. 4. Speed of convergence as a function of depth.

Figure 5b shows the result obtained with the iterative paraperspective algorithm in the case of lines, described in this paper; In figure 5c, we used a non linear method (Levenberg-Marquardt algorithm) from line correspondences.

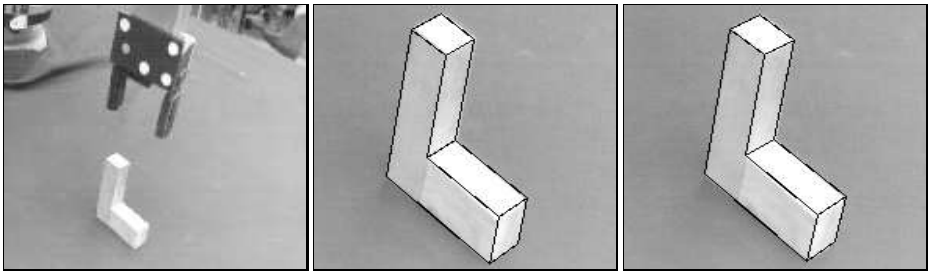


Fig. 5. (a–c) An example of applying the object pose algorithms.

7 Discussion and perspective

In this paper, we developed two iterative algorithms (iterative weak perspective and iterative paraperspective algorithms) to deal with 3-D to 2-D line correspondences, for a non coplanar or coplanar object model. The two algorithms have the same accuracy. However, it's better to use the weak perspective algorithm for line correspondences because computation cost is lower. We only need to compute one inverse matrix which remains the same at each iteration to solve the linear system (equation (10)). Moreover, The orientation of the camera (**i**, **j**, **k**) is more easily computed with the weak perspective algorithm.

References

1. S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104, November 1996.
2. D. Dementhon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, 1995.
3. M. Dhome, M. Richetin, J.T. Lapresté, and G. Rives. Determination of the attitude of 3D objects from sigle perspective view. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, 1989.
4. P. Gros. Matching and clustering: Two steps towards automatic model generation in computer vision. In *Proceedings of the AAAI Fall Symposium Series: Machine Learning in Computer Vision: What, Why, and How?, Raleigh, North Carolina, USA*, pages 40–44, October 1993.
5. R. Horaud, S. Christy, F. Dornaika, and B. Lamiroy. Object pose: Links between paraperspective and perspective. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 426–433, Cambridge, Mass., June 1995. IEEE Computer Society Press.
6. D. Oberkampf, D.F. Dementhon, and L.S. Davis. Iterative pose estimation using coplanar feature points. *Computer Vision, Graphics and Image Processing*, 63(3):495–511, May 1996.
7. T.Q. Phong, R. Horaud, A. Yassine, and P.D. Tao. Object pose from 2D to 3D point and line correspondences. In *International Journal of Computer Vision*, pages 225–243. Kluwer Academic Publishers, July 1995.
8. C.J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In J.O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, pages 97–108, May 1994.
9. J.S.C. Yuan. A general phogrammetric solution for the determining object position and orientation. IEEE *Transactions on Robotics and Automation*, 5(2):129–142, 1989.